ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

«САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ, МЕХАНИКИ И ОПТИКИ»

Факультет безопасности информационных технологий

Кафедра проектирования и безопасности компьютерных систем

Дисциплина:

«Операционные системы»

ОТЧЕТ О ВЫПОЛНЕНИИ

ЛАБОРАТОРНОЙ РАБОТЫ №7

Выполнил:

N3247 Гаврилова В.В.

Проверил:

Ханов А. Р.

Санкт-Петербург

2021г.

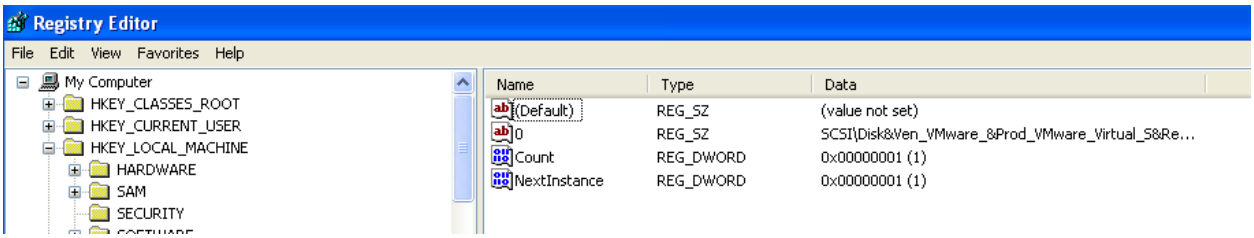
Способ 1 (Windows)

Если посмотреть идентификатор жесткого диска в диспетчере устройств на виртуальной машине, то в его составе можно увидеть подобные строчки:

1. DiskVirtual для VirtualPC
2. DiskVBOX_HARDDISK для Virtual Box
3. Prod_VMware_Virtual для VMware Workstation”

Самый простой способ узнать наименование жесткого диска — прочитать значение ключа с именем «0» в ветке реестра HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Disk\Enum. В этом месте перечисляются все дисковые накопители в системе, и первым, как раз в ключе с именем «0», будет тот диск, с которого произошла загрузка системы.

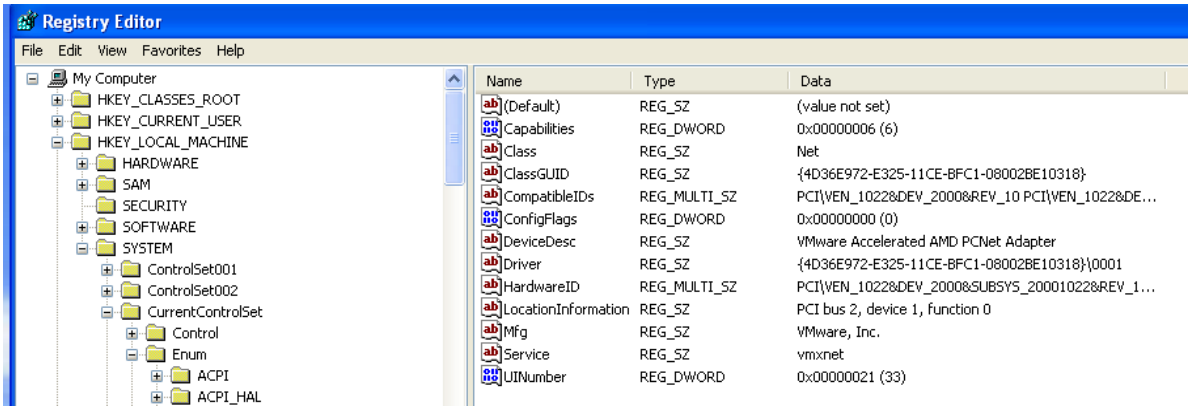
Пример для VmWare:



Способ 2 (Windows)

Данные о видеоадаптере можно увидеть в HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\PCI. В этой ветке перечислено все, что подключено к шине PCI, в том числе и видеокарта.

1. Для VirtualPC это строчка вида VEN_5333&DEV_8811&SUBSYS_00000000&REV_00, которая определяет видеоадаптер S3 Trio 32/64, эмулируемый виртуалкой от Microsoft
2. Для VirtualBox видеокарта описана последовательностью VEN_80EE&DEV_BEEF&SUBSYS_00000000&REV_00, что расшифровывается как «VirtualBox Display»
3. у Parallels Workstation — строка VEN_1AB8&DEV_4005&SUBSYS_04001AB8&REV_00 определяет видеоадаптер «Parallels Display».
4. У VmWare Workstation это выглядит следующим образом:



Способ 3

Использование MAC-адреса для идентификации производителя сетевой карты - не самый надежный способ (ибо MAC-адрес довольно-таки просто поменять), но тем не менее его можно использовать для детектирования виртуальных машин в качестве дополнительной проверки.

Поскольку первые три байта MAC-адреса сетевой карты определяют ее производителя, мы можем легко прочитать это значение с помощью API-функции GetAdaptersInfo (BM Windows, библиотека iphlpapi.dll) либо выведя MAC-адреса адаптеров из /sys/class/net/*/address в Linux.

Пример для Linux:

Список адресов, соответствующих различным производителям виртуальных машин:

1. VMware (VMware Workstation) 00:05:69 00:0c:29 00:1c:14 00:50:56
2. Microsoft (Virtual PC) 00:03:ff 00:0d:3a 00:50:f2 7c:1e:52 00:12:5a 00:15:5d 00:17:fa 28:18:78 7c:ed:8d 00:1d:d8 00:22:48 00:25:ae 60:45:bd Dc:b4:c4
3. Oracle (VirtualBox) 08:00:20
4. Parallels (Parallels Workstation) 00:1c:42

Соответственно на скриншоте можно видеть, что второй адаптер eth0 имеет первых три байта 00:0c:29, а значит, что виртуалка запущена из-под VmWare, что правда.

Спососб 4

Для нормальной работы практически все виртуальные машины требуют установки дополнений к гостевой операционной системе, например

1. VirtualBox VBoxTray.exe VBoxService.exe
2. Parallels Workstation prl_cc.exe prl_tools.exe SharedIntApp.exe
3. Virtual PC vmusrvc.exe vmsrvc.exe
4. VMware Workstation vmtoolsd.exe

Без этих дополнений работа с виртуальной машиной довольно затруднительна. Соответственно, возможно найти эти процессы в трее с помощью функций: CreateToolhelp32Snapshot, Process32First и Process32Next для Windows.

В линукс же просто:

Способ 5 (Windows)

Помимо признаков наличия специфического оборудования, в реестре можно увидеть и другие следы, оставляемые виртуальными машинами. Некоторые из них базируются в ветке HKEY_LOCAL_MACHINE\HARDWARE\ACPI\DSDT. Достаточно в этом месте проверить наличие таких ключей:

1. VirtualBox VBOX__
2. Parallels Workstation PRLS__
3. Virtual PC AMIBI
4. VMware Workstation PTLTD__

Способ 6 (Windows)

Описан подробно в статье [1].

Как было отмечено выше, устройства виртуальных машин имеют уникальные идентификаторы DeviceID, VendorID, SubsystemID и Subsystem Vendor ID в конфигурационном пространстве. Порт адреса задаёт шину, устройство и адрес регистра в конфигурационном пространстве устройства. Суть метода: число шин вычисляется подсчётом мостов PCI-to-PCI, после чего непосредственно делаем проход по устройствам (Если в ответ приходит 0FFFFh – то устройства не существует) и получаем значения обозначенных выше идентификаторов.

Исходныĕ код на языке Delphi для получения необходимых идентификаторов приведён ниже.

```
Function IsDevice(Bus,Dev,Reg,Fun
:Byte):DWORD;
var
Addr : DWORD;
Temp : DWORD;
Resultat : DWORD;
Begin
GetPortVal($0CF8, Temp, 4);
Addr := (1 shl 31) + (Bus shl 16) +
(Dev and $1F) shl 11 +
(Fun and $07) shl 8 + (Reg and $3F)
shl 2;
// Пишем в порт адрес
SetPortVal($0CF8, Addr, 4);
GetPortVal($0CFC, Resultat, 4);
// Возвращаем что было
SetPortVal($0CF8, Temp, 4);
Result := Resultat;
End;
```

Схема обхода представляет собоӥ три вложенных цикла по параметрам:

- Bus = 00h...FFh;
- Dev = 00h...1Fh;
- Fun = 00h...07h.

Обход повторяется два раза, в первый раз производим подсчёт количества мостов N PCI-to-PCI. Таким образом, во втором обходе Bus=00h...Nh.

Есть у этого подхода и минусы – начиная с Vista прямой доступ к портам невозможен, нужно писать драйвер.

Способ 7 (Windows)

Помимо вспомогательных процессов можно анализировать и объекты, порождаемые этими процессами в ОЗУ компьютера

Анализ объектов вспомогательных приложений

VirtualBox	DeviceVBoxMiniRdrDN, DeviceVBoxGuest
Parallels Workstation	Deviceprl_pv, Deviceprl_tg, Deviceprl_time, DevicePrlMemDev
Virtual PC	DeviceVirtualMachineServices DeviceVirtualMachineServicesPCI DeviceVirtualMachineServicesUSB
VMware Workstation	DevicePrlMemDevPci, DevicePrlMemDev

Идентификация виртуальной машины в этом методе основана на попытке создания объекта с именем, приведенным в таблице 2. Данная операция осуществляется при помощи функции WinAPI CreateFile. Ошибка создания объекта является признаком существования объекта. Обойти подобную проверку возможно перехватом вызова функции CreateFile или непосредственной модификацией системной динамической библиотеки Microsoft Windows - kernel32.dll.

Способ 8

Анализ времени выполнения инструкций: например, cupid (лакмусовая бумажка, запрашиваем сведения о процессоре)

Код на C:

```
  GNU nano 5.2                                          asm_test2.c
#include <stdio.h>

extern unsigned _ticks_per_cpuid();

int main(){
        printf("ticks per cpuid: %u\n", _ticks_per_cpuid());
        return 0;
}
```

Код функции на asm x86_64 (nasm):

```
  GNU nano 5.2
BITS 64

section .text

global _ticks_per_cpuid

_ticks_per_cpuid:
        rdtsc
        mov edi, eax
        xor rax, rax
        cpuid
        rdtsc
        sub eax, edi
ret
```

Тестируем:

```
root@kali:/mnt/hgfs/vm_share/labs/OS/lab7# nasm -felf64 ticks.asm -o ticks_elf.o
root@kali:/mnt/hgfs/vm_share/labs/OS/lab7# gcc -Wall -g -c asm_test2.c -o asmtest2.o
root@kali:/mnt/hgfs/vm_share/labs/OS/lab7# gcc -o asmtest2 asmtest2.o ticks_elf.o
root@kali:/mnt/hgfs/vm_share/labs/OS/lab7# ./asmtest2
ticks per cpuid: 1622
root@kali:/mnt/hgfs/vm_share/labs/OS/lab7# ./asmtest2
ticks per cpuid: 1598
root@kali:/mnt/hgfs/vm_share/labs/OS/lab7# ./asmtest2
ticks per cpuid: 1504
```

Собираем то же самое под хост Windows 10 и тестируем:

```
E:\vm_share\labs\OS\lab7>gcc -o test.exe asm_win.o ticks_win.o

E:\vm_share\labs\OS\lab7>test.exe
ticks per cpuid: 112

E:\vm_share\labs\OS\lab7>test.exe
ticks per cpuid: 112

E:\vm_share\labs\OS\lab7>test.exe
ticks per cpuid: 114
```

Пожалуйста: команда cupid исполняется медленно под гипервизором, так как ему нужно время на обертку инструкции до ее исполнения.

Способ 9

Последний способ выявления виртуализации – «атака» RedPill (ну не атака, метод).

Операционной системе для работы необходим доступ к так называемой таблице векторов прерываний. Начиная с процессора 80286, адрес в физической памяти этой таблицы хранится в 48-битном регистре IDTR.

Поскольку регистр Interrupt Descriptor Table Register находится под контролем операционной системы, то создаётся альтернативная таблица, с которой работает гипервизор. Для этой новой таблицы выделяется отдельное место в памяти с более старшим адресом, чем у оригинальной таблицы, и этот более старший адрес записывается в IDTR. Йоанна Рутковская обнаружила, что адрес IDTR с старшим байтом более 208 (или D0 в шестнадцатеричной системе) указывал на факт перезаписи IDTR более старшим адресом для таблицы прерываний, то есть сравнив значение старшего байта указателя из IDTR с D0 можно было сделать вывод о присутствии гипервизора. Значение указателя из IDTR считывалось при помощи команды SIDT (0f010f <адрес, куда возвращать адрес IDTR>).
Код на С:

```
  GNU nano 5.2
#include <stdio.h>

int Neo_picks_the_RedPill(){
        unsigned char m[6];
        unsigned char redPill[] = "\x0f\x01\x0d\x00\x00\x00\x00\xc3";
        *((unsigned*)&redPill[3]) = (unsigned)m;
        ((void(*)())&redPill)();
        if (m[5] > 0xd0) return 1;
        else return 0;
}

int main(){
        if (Neo_picks_the_RedPill) printf("we are in matrix, Morpheus\n");
        else printf("Pls, Neo, stop taking drugs\n");
        return 0;
}
```

```
root@kali:/mnt/hgfs/vm_share/labs/OS/lab7# gcc redpill.c -o rp
redpill.c: In function ‘Neo_picks_the_RedPill’:
redpill.c:6:36: warning: cast from pointer to integer of different size [-Wpointer-to-int-cast]
    6 |         *((unsigned*)&redPill[3]) = (unsigned)m;
      |                                    ^
root@kali:/mnt/hgfs/vm_share/labs/OS/lab7# ./rp
we are in matrix, Morpheus
```

Ссылка на статью:

[1] ОПРЕДЕЛЕНИЕ ЗАПУСКА ИЛИ РАБОТЫ ПРИЛОЖЕНИЯ В ВИРТУАЛЬНОЙ МАШИНЕ НА ОСНОВЕ АНАЛИЗА КОНФИГУРАЦИОННОГО ПРОСТРАНСТВА PCI УСТРОЙСТВ. Черемнов А. Г.